Naval Research Laboratory
Washington, DC 20375-5000

NRL Memorandum Report 6788

Digital Beamforming and Pulse Compression in an Adaptive Array Radar System

FREDERICK W. LEE AND M. BURROUGHS

Airborne Radar Branch
Radar Division

February 25, 1991

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1991 February 25	

4. TITLE AND SUBTITLE

Digital Beamforming and Pulse Compression in an Adaptive Array Radar System

5. FUNDING NUMBERS

PE - 62111N
PR - RA11W91
JO - 53-0662-0-0

6. AUTHOR(S)

Frederick W. Lee and M. Burroughs

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, S.W.
Washington, DC 20375-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL Memorandum Report 6788

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NRL/NADC
Warminster, PA 18974-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

New digital techniques for forming antenna beam patterns and matched filtering of radar waveforms are investigated and compared to the traditional analog methods. The development at NRL of an integrated digital processor for performing both adaptive beamforming and pulse compression is presented along with performance data obtained from experiments performed in the Adaptive Processing Laboratory.

14. SUBJECT TERMS

15. NUMBER OF PAGES
37

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

CONTENTS

DIGITAL BEAMFORMING AND PULSE COMPRESSION IN AN ADAPTIVE ARRAY RADAR SYSTEM

I. INTRODUCTION

The purpose of this investigation was to develop and demonstrate the capabilities of a digital signal processing (DSP) system to perform real-time adaptive beamforming and matched filtering of post received signals in the Adaptive Processing Laboratory. Previously, the Adaptive Processing Laboratory employed a digitally controlled, analog signal processor to perform the required beamforming task and surface acoustic wave (SAW) dispersive delay lines to match filter the encoded radar waveforms. As reported earlier[1], the Intermediate Frequency (IF) Beamformer was suited only for closed loop adaptive algorithms, failing to successfully implement open loop algorithms. This failure was attributed to the inability of the IF Beamformer to accurately "weight" the IF signals prior to the summing network. The principal cause of these errors was due to the analog nature of the complex phase modulators that were used as the weighting devices.

Another weakness of the analog system was the interchannel mismatch due to the SAW dispersive delay lines. The delay lines are typically matched at a 33 dB level, however, channel match on the order of 40 dB can be achieved with the delay lines replaced by non dispersive lump constant (LC) filters. This then requires that the expanded radar waveforms be compressed after the digital beamforming network.

Section II of this report will briefly overview the Adaptive Processing Laboratory equipment and capabilities. Section III will give a detailed description of the four digital subsystems that comprise the digital beamformer and pulse compressor. Section IV will detail the control and analysis software developed and section V will present the results of the analysis of both the digital beamforming capabilities and compressed pulse characteristics.

II. SYSTEM OVERVIEW

Most of the equipment in the Adaptive Processing Laboratory was part of a flight test system jointly developed by NRL and General Electric under contract number N00173-77-C-0283. In general this system is an eight element ultra-high frequency (UHF) linear array radar. The primary purpose of the flight test system was to gather clutter and ECM data in an airborne environment aboard an NRL P-3A aircraft. The components of the flight test system utilized in the Adaptive Processing Laboratory were the receiver group, portions of the modified flight test radar, and the NRL recording and control group.

The receiver group consists of eight identical channels, each channel having its own receiver, IF amplifier, and synchronous demodulator. The receiver units provide amplification and frequency conversion of the received UHF signals, converting them to the 75 MHz IF frequency. The IF amplifiers provide bandpass filtering using lump constant (LC) filters. Additionally, digital gain control of the amplification is also available in this unit. Finally, the synchronous demodulator converts the uncompressed IF output using the 75 MHz coherent oscillator (COHO) to baseband inphase (I) and quadraturephase (Q) video.

Manuscript approved December 21, 1990.

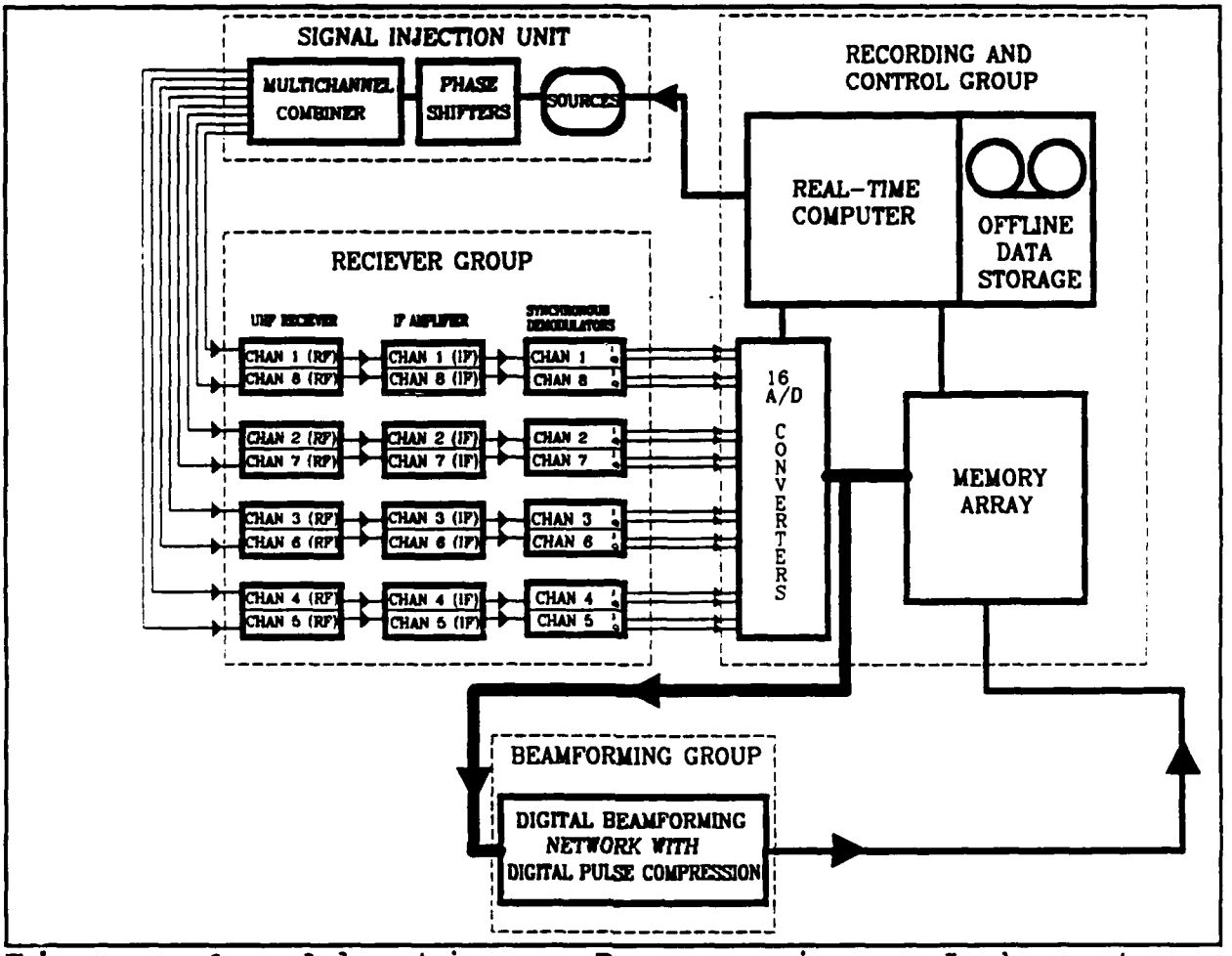

Figure 1 Adaptive Processing Laboratory Block Diagram

The portion of the flight test radar required for this experiment was the WRA 9 pulse and signal generator. The purpose of this unit is to provide timing signals and oscillators for the entire adaptive system as well as generate the test targets and channel balance test pulse. This unit is functionally divided into eight separate elements (Figure 2). The primary element is the timing coherent master oscillator (A2 TMG COMO). This 15 MHz crystal oscillator is the heart of all coherent operation in the system.

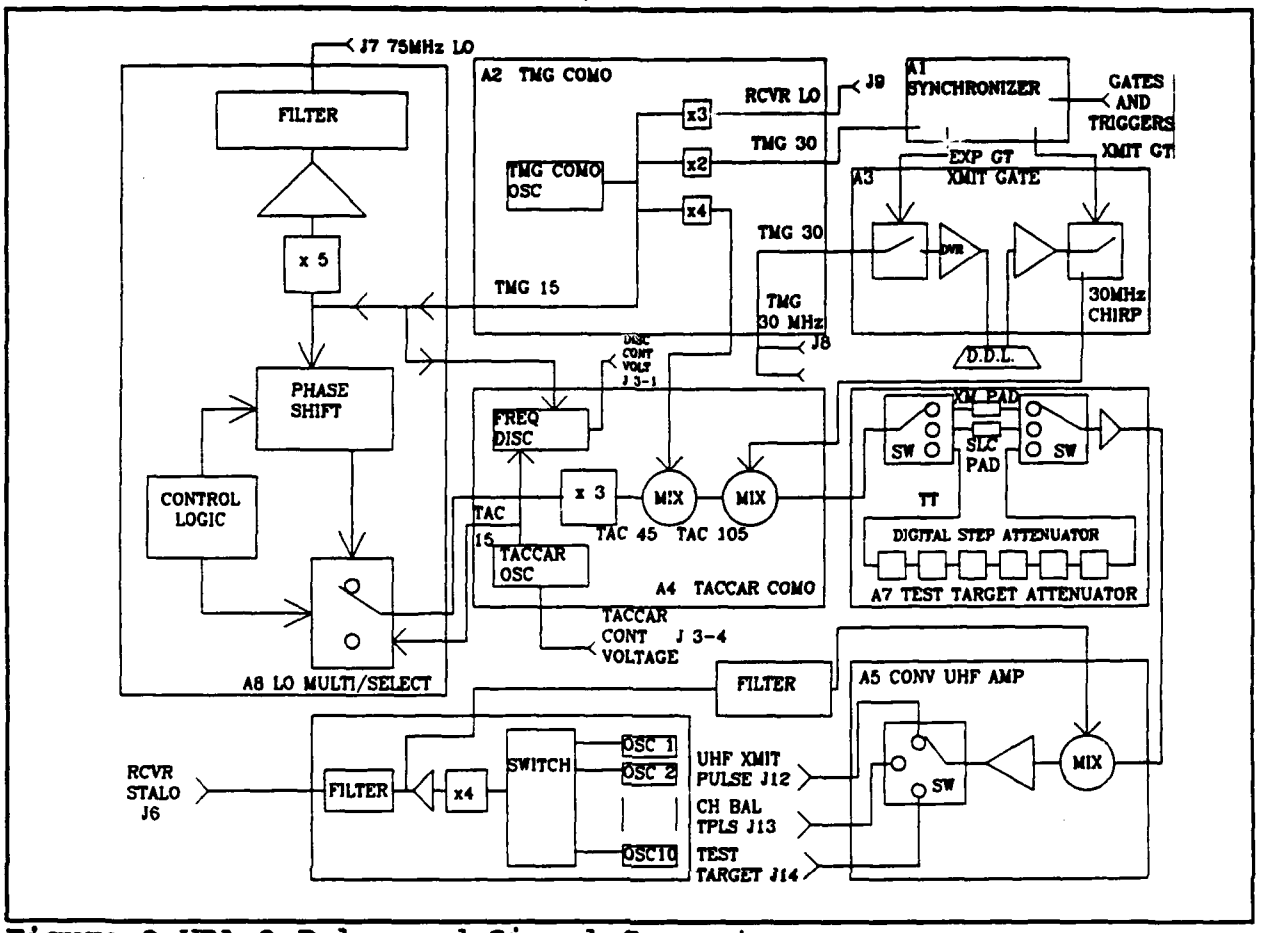

Figure 2 WRA 9 Pulse and Signal Generator

It is multiplied in the local oscillator multiplier/select (A8 LO MULTI/SELECT) to create the 75 MHz COHO. The COMO signal is also doubled to 30 MHz and sent to the transmit gate (A3 XMIT GATE), which is controlled by the synchronizer (A1), to generate the chirp waveforms used by the converter UHF amplifier (A5). The converter mixes the IF chirp waveforms with the stable local oscillator (A6 STALO) to generate the UHF transmit pulse, channel balance test pulse, and test targets. The STALO is then sent to the receivers to provide for coherent down conversion of the received signals.

The NRL recording and control group (Figure 3) is primarily a data acquisition system optimized for acquiring radar data. The baseband I and Q video outputs of the synchronous demodulators in the receiver group are fed in parallel to a bank of 16 analog-to-digital (A/D) converters. The Computerlab 10 bit A/D converters are clocked by the 15 MHz COMO divided down to a 5 MHz sample rate. The digitized signals are transmitted over a 160 bit wide data bus to a high speed bulk memory system. The Microram 3000N memory system is used to buffer the data at the full data rate of nearly 100 megabytes per second and, thru a custom interface, allows control over how the data is collected and sent to the Rolm 1602 computer. The data can be collected as contiguous range cells, pulses, or a combination of both. The computer can then retrieve the data for any particular radar channel or for all channels. Once in the computer the data

can either be recorded on magtape, displayed, or processed.

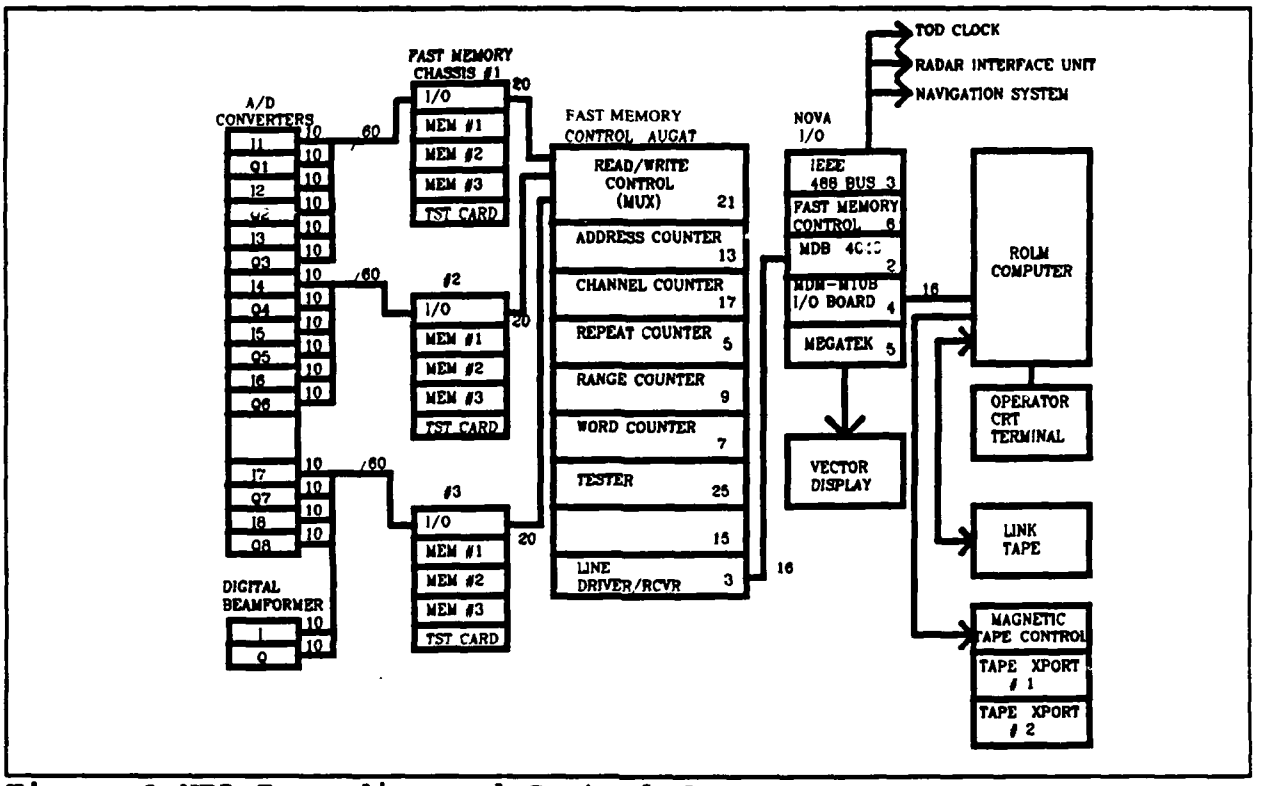

Figure 3 NRL Recording and Control Group

Due to logistical problems, the transmitter and antenna used during the flight test were not incorporated into the laboratory equipment. A signal injection unit (see ref. 1 for detail) was built to emulate target returns in the presence of multiple mainlobe and sidelobe interference sources.

Initially, the beamforming network for the flight test system was a tapered set of fixed attenuators providing a fixed look angle with a 25 dB Dolph-Tschebyscheff beam pattern. It was desired for the laboratory system to have a more versatile adaptive beamformer, consequently a digitally controlled intermediate frequency (IF) beamformer was developed. This analog signal processor was capable of setting arbitrary look angles and was well suited for closed loop adaptive algorithms. However, the IF beamformer was unable to implement open loop adaptive algorithms. The development of the DSP real-time adaptive beamformer solved this problem.

III. DIGITAL SUBSYSTEM DESCRIPTION

The DSP real-time adaptive beamforming system can be functionaly divided into four digital subsystems. These are the:

1. Digital complex phasor modulators (DCPM).
2. Inphase (I) and quadphase (Q) accumulators.
3. Digital pulse compression filter (DPCF).

4. Microprocessor based filter controller.

A data flow diagram (Figure 4) shows the interaction and data paths of each of the subsystems. A detailed description of each of the four digital subsystem follows.

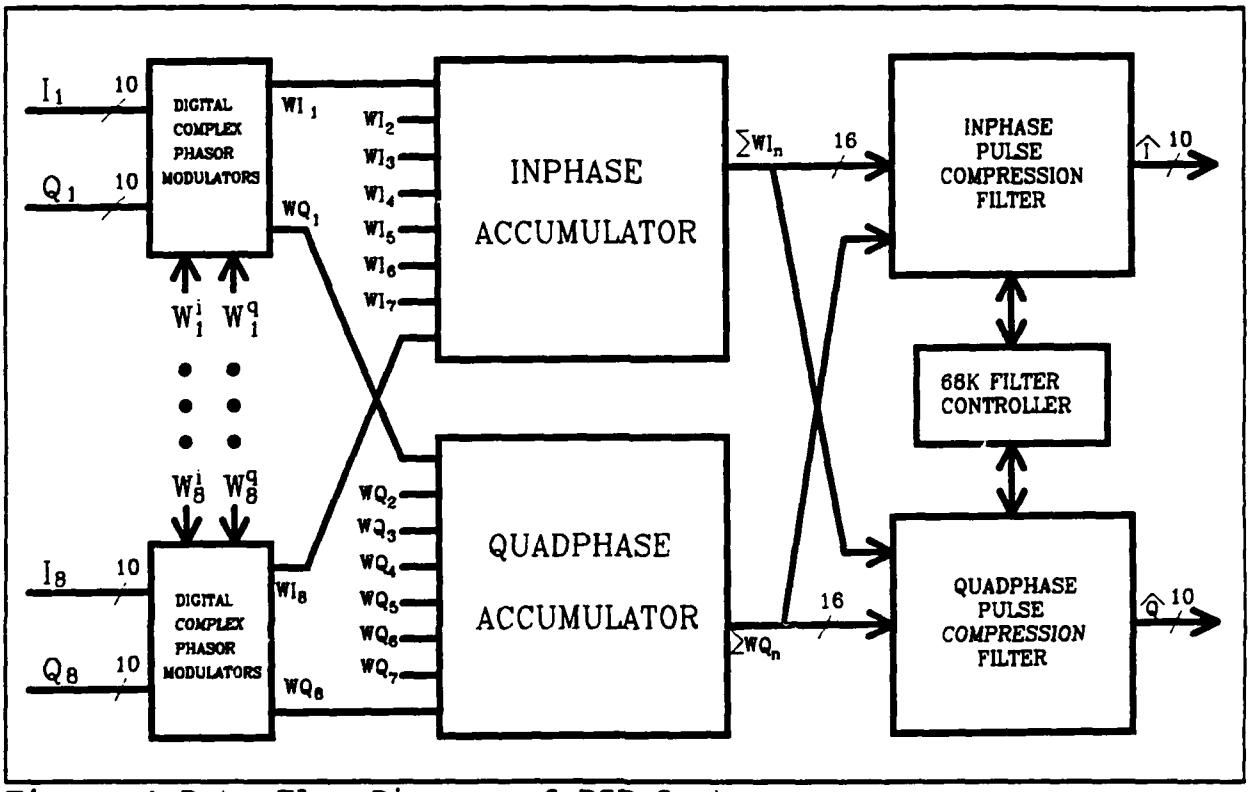

Figure 4 Data Flow Diagram of DSP System

III A. DIGITAL COMPLEX PHASOR MODULATORS

The DCPM is a digital signal processing card used to simultaneously control the phase and amplitude of an orthogonal pair of sampled analog voltages. The DCPM performs this function by simply weighting the sampled voltages, designated I for in phase and Q quadrature phase, by multiplying them by a complex weight W.

$$V = I + j\ Q \tag{3-1}$$

$$W = W_i + j\ W_q \tag{3-2}$$

$$F = V * W = (IW_i - QW_q) + j\ (IW_q + QW_i) \tag{3-3}$$

Writing V and W in exponential form and solving for F yields the relationship between the complex weight and the amplitude and phase changes that are applied to the original sampled signal.

$$V = (I^2 + Q^2)^{1/2}\ e^{j(ATAN(Q/I))} \tag{3-4}$$

$$W = (W_i^2 + W_q^2)^{1/2}\ e^{j(ATAN(Wq/Wi))} \qquad (3\text{-}5)$$

$$F = (I^2 + Q^2)^{1/2} * (W_i^2 + W_q^2)^{1/2}\ e^{j(ATAN(Q/I) + ATAN(Wq/Wi))} \qquad (3\text{-}6)$$

The DCPM is principaly comprised of four Analog Devices ADSP-1012 12 x 12 bit CMOS digital multipliers (Figure 5), one for each of the inner products ($I*W_i$ $Q*-W_q$ $I*W_q$ $Q*W_i$) depicted in the rectangular representation of the output F of the complex multiplication. The ADSP-1012 is TTL compatible with separate input data buses and registers for each of the two multiplicands as well as a separate output bus and register for the product. The ADSP-1012 is capable of performing two's complement multipications with the ability to internally round the product output from its full precision of 24 bits to 12 bits.

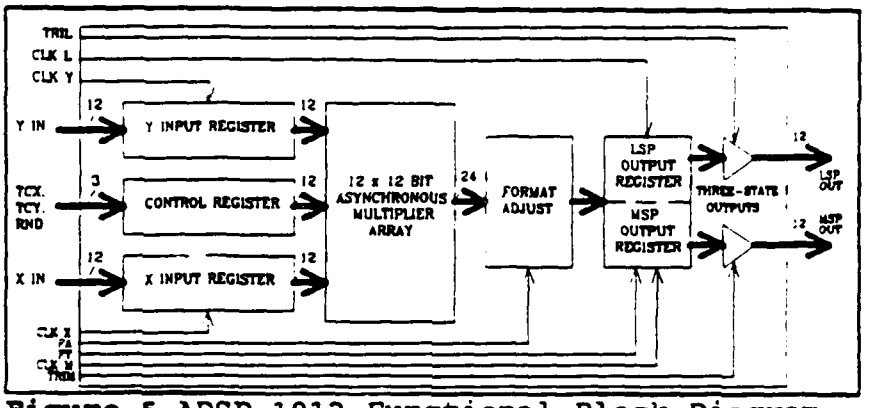

Figure 5 ADSP-1012 Functional Block Diagram

Referring to Figure 6, the DCPM has two 24 bit input buses, one synchronous, the other asynchronous and one 26 bit output bus. The synchronous bus, denoted the Complex Data Input Bus (CDIB), is further subdivided into two 12 bit data buses that are driven by the I and Q sampled voltage outputs from the data aquisition system A/D converters. Since each A/D converter has only 10 bits of resolution, the outputs are left justified to 12 bits upon input to the DCPM, effectively multiplying the input data by a factor of 4. The CDIB is clocked at the A/D converters 5 MHz sample rate using the Input Data Valid (IDV) signal, a strobe generated on the DCPM board from the A/D data ready pulses.

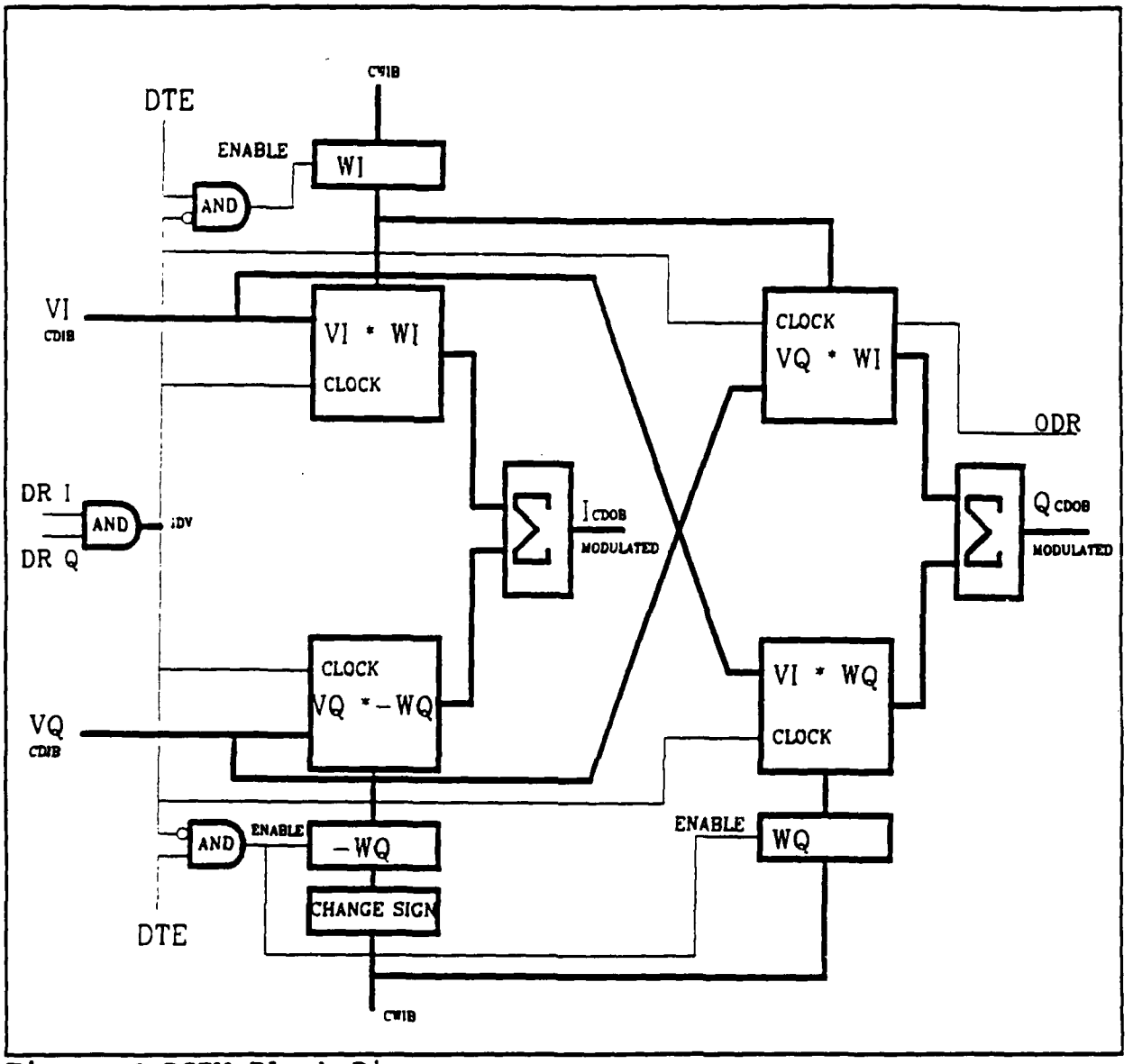

Figure 6 DCPM Block Diagram

The asynchronous bus, the Complex Weight Input Bus (CWIB), is similar to the CDIB in that it is also subdivided into two 12 bit data buses, one each for the W_i and W_q data words that comprise the complex weight. However, the CWIB being an asynchronous bus, transfers data from the host computer interface at an irregular rate determined by the particular algorithm calculating the new weight. This update is initiated by the host computer asserting the Data Transfer Enable (DTE) strobe. The output bus, referred to as the Complex Data Output Bus (CDOB), is the mathematical complex product of the input data and weight. The CDOB is also a compound bus comprised of two 13 bit data buses to handle the real and imaginary parts of the output. This bus is accompanied by an Output Data Ready (ODR) strobe which is coherent with the IDV strobe. Figure 7 shows the timing relationships between the data and clocks of the synchronous data path. Note that the DTE can be asserted at any time, however, the data transfer (DT) will not

take place until the IDV strobe goes low.

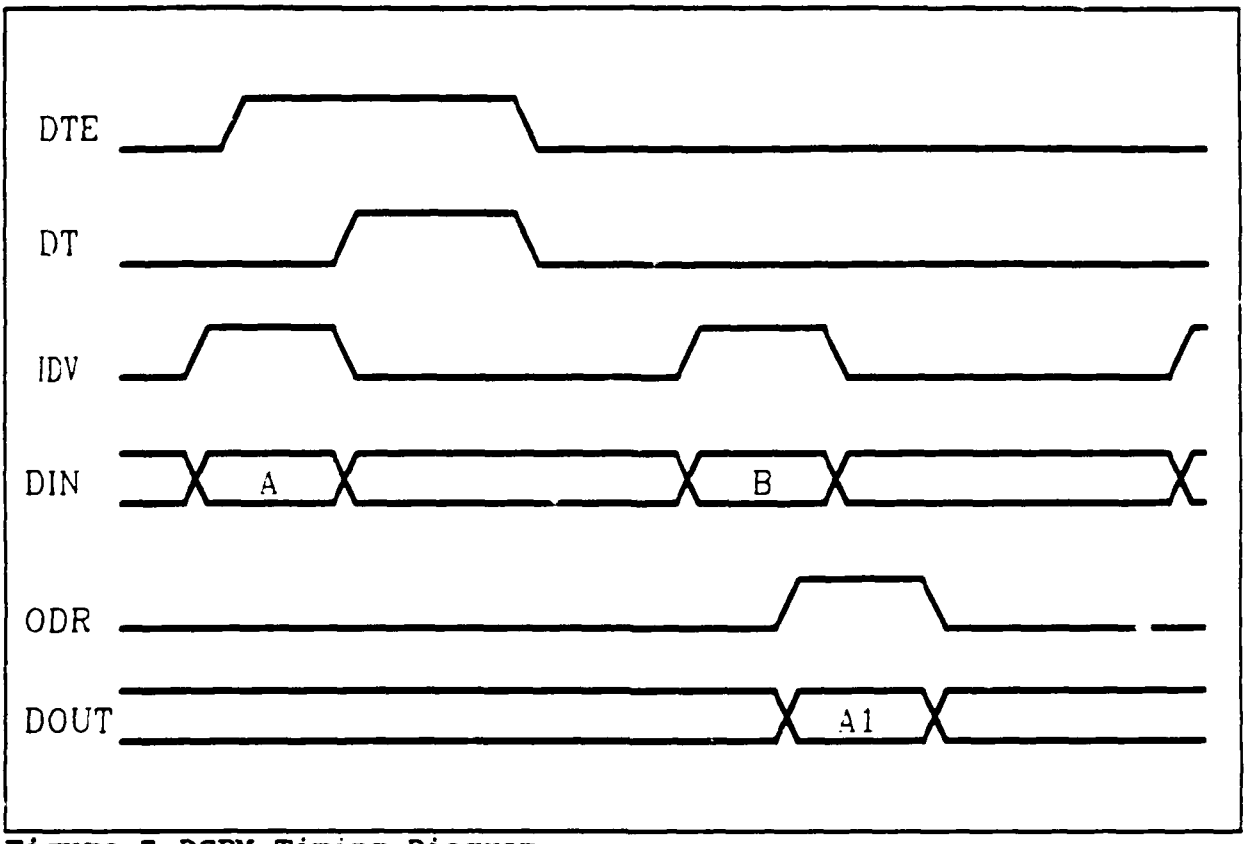

Figure 7 DCPM Timing Diagram

Functionally the DCPM is quite simple. Starting with a new weight present on the CWIB, the first action to occur is that the W_q data word is negated. For a two's complement number this is accomplished by inverting the data bits and then adding '1'. This all transpires prior to the new weight being written into the $-W_q$ register on the DCPM. After the DTE strobe is asserted and the IDV is low, the new weight on the CWIB, as well as the $-W_q$ calculated from it, are written into there primary DCPM registers, ready to be clocked into their respective ADSP-1012's internal Y input register on the rising edge of IDV. The IDV, as stated before, is generated on board the DCPM using the A/D data ready strobes. The IDV goes high when the input data from the A/D converters has had time to set up on the inputs to the X input register internal to the ADSP-1012. The IDV is then sent to a delay generator to create the ODR strobe used to clock the output 130 nsec after the multiplication has been initiated by IDV. This corresponds to the max t_{MC} or clocked multiply time[2] of the ADSP-1012. The remaining 70 nsec is used to allow the cross term additions to take place prior to the next multiplication cycle being initiated. This introduces a one clock pipeline delay thru the DCPM with the output of the previous multiplication valid and stable on the CDOB prior to the rising edge of the ODR strobe. This allows the ODR to be used to latch the output of the DCPM into the input registers of the Inphase (I) and Quadphase (Q) accumulators.

III. B. I AND Q ACCUMULATORS

The I and Q accumulators are two identical DSP cards, differing only in where their inputs originate. Functionally, the accumulator sums in one 5 Mhz clock cycle eight individual inputs. This sum S is decribed mathematically as:

$$S = \sum F_n \tag{3-7}$$

It accomplishes this accumulation in just a single clock cycle by using a tree adder structure (Figure 8). Using this structure, the contents of the input registers 1 thru 8 are initially summed in pairs (R1+R2,R3+R4,R5+R6,R7+R8). The output of these partial sums are in turn added in pairs, ((R1+R2)+(R3+R4),(R5+R6)+(R7+R8)), until finally these outputs are summed and latched into the output register R_{out} such that:

$$R_{out} = (((R1+R2)+(R3+R4))+((R5+R6)+(R7+R8))) \tag{3-8}$$

The input registers of the I and Q accumulators are driven by the respective real and imaginary parts of the complex data output buses from each channel's DCPM. If in equation (3-3) were expressed as a vector F_n such that:

$$F_n = V_n * W_n \tag{3-9}$$

and $(V_n * W_n)$ is substituted into equation (3-7), the resulting equation:

$$S = \sum V_n * W_n \tag{3-10}$$

is the vector dot product of the sampled complex voltage vector V_n and the complex weight vector W_n. This weighted sum is updated for each sample period of the A/D converters and constitutes the digital beamforming function of the DSP real-time adaptive beamforming system.

III C. DIGITAL PULSE COMPRESSION FILTER

The Digital Pulse Compression Filter (DPCF) is comprised of a pair of digital signal processing cards used to digitally match filter the I and Q outputs of the digital beamformer. Pulse compression is a technique which allows a radar to transmit a long pulse while simultaneously obtaining the range resolution of a short pulse. The concept requires an encoded transmit pulse and a filter mechanism at the receiver. One method of implementation uses linear frequency modulation to encode the transmit pulse and a matched filter in the receiver[3]. A simple graphic illustration, from Cook's paper, of this process follows.

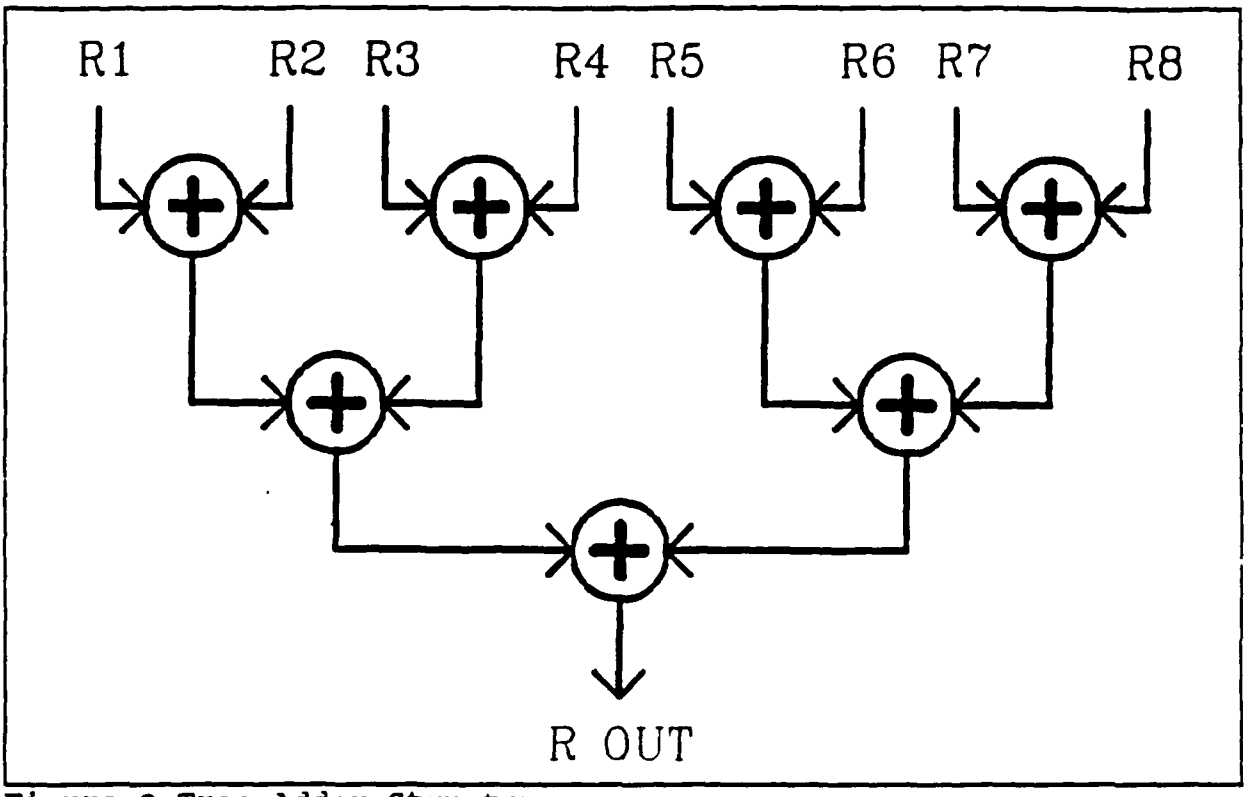

Figure 8 Tree Adder Structure

A transmit pulse, of width $2T_o$, with its FM characteristics is shown in (Figure 9.a). The time vs. frequency characteristics of this pulse are shown in (Figure 9.b).

The matched filter's time delay vs. frequency characteristics are shown in (Figure 9.c). When the received pulse enters the filter, the high frequency end of the pulse undergoes the greatest delay while the low frequency end of the pulse is delayed the least. Ideally, the energy in the high frequency end is slowed just enough so that it emerges form the filter along with that from the low frequency end of the pulse (Figure 9.d).

The amplitude of the compressed pulse is directly proportional to the ratio of the uncompressed pulse width, $2T_o$, to the compressed pulse width. The theory required to implement this technique is presented next.

Filtering in the continuous time domain may be expressed as a convolution between the filter's impulse response, h(t), and its input, x(t). Thus, the filter's output, y(t), is·

$$y(t) = \int h(t-z)\, x(z)\, dz. \qquad (3\text{-}11)$$

When x(t) is sampled, the system becomes discrete (actually digital) in

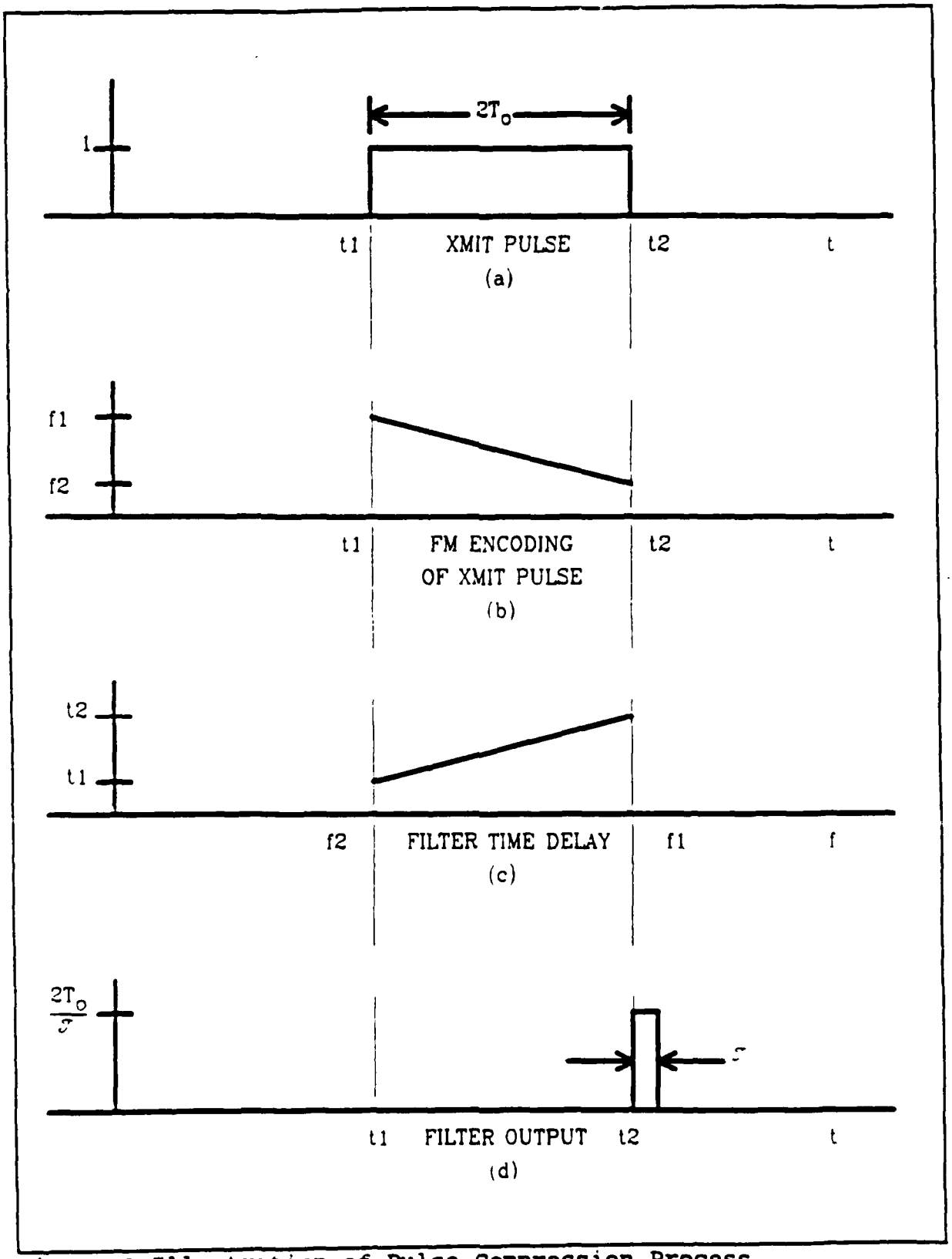

Figure 9 Illustration of Pulse Compression Process

time and the integral above becomes a sum. Therefore:

$$y[nT] = \sum h[(n-k)T]\ x[kT] \tag{3-12}$$

describes the output of the digital filter system. The variable 'T' is usually fixed for a given system and is equal to $1/f_s$, the sampling frequency. By application of the Fourier transform, the convolution in time becomes a product in frequency space so that:

$$Y_s(\omega) = \sum H_s(\omega)\ X_s(\omega). \tag{3-13}$$

Here, $X_s(\omega)$ is continuous function of frequency[4], periodic in ω_s. This signal is a replicated, scaled version of the Fourier transform of the continuous signal, x(t), i.e.,

$$X_s(\omega) = 1/T \sum X(\omega)\ d(\omega-k\omega_s) \tag{3-13}$$

$$X_s(\omega) = 1/T \sum X(\omega-k\omega_s). \tag{3-14}$$

It should be noted that both $H_s(\omega)$ and $Y_s(\omega)$ are periodic in ω_s, as well.

The work above has shown that a sampled data system will yield a result that contains the same information as the continuous system from which it was derived. Thus, a digital filter, if it may be synthesized, will provide a valid output equivalent to that from its analog counterpart. To derive an expression for the coefficients for a digital pulse compression filter consider first the received pulse at baseband, before sampling. It may be represented as x(t). To implement a matched filter design, the impulse response of the compression filter, h(t), must be the complex conjugate of the ideal received pulse, x(t). Again invoking the Fourier transform, this may be represented as:

$$H_s(\omega) = X_s^*(\omega) \tag{3-15}$$

$$= 1/T \sum X^*(\omega-k\omega_s) \tag{3-16}$$

The impulse function represented by this frequency function is of infinite length. In order to realize this filter using a non-recursive structure (Finite Impulse Response, or FIR filter), the series must be truncated to a finite number of terms, N[5]. (N is also known as the length of a FIR filter or the number of taps in the filter.) The frequency function approximating the transform of the desired impulse response is now:

$$H_{FIR}(\omega) = 1/T \sum X^*(\omega-k\omega_s). \tag{3-17}$$

Since $H_{FIR}(\omega)$ is periodic in ω_s, it may be represented as a Fourier series. Thus, with a change in summation index, $H_{FIR}(\omega)$ may be written as:

$$H_{FIR}(\omega) = 1/T \sum c_k\ e^{j\omega kT}. \tag{3-18}$$

Solving for the complex Fourier coefficients yields

$$c_k = 1/\omega_s \int H_{FIR}(\omega)\ e^{-j\omega kT}\ d\omega \tag{3-19}$$

or

$$c_k = 1/2 \int X^* (\omega - k\omega_s)\ e^{-j\omega kT} d\omega. \qquad (3\text{-}20)$$

Once these coefficients are known, the weights for the time domain filter are easily determined. This is readily seen by observing that the time function corresponding to transform above is just the sum of N multiplies delayed in time by the sample period.

To build a filter with ICs that can only handle real (non-complex) values, the above equation must be broken into its real and imaginary component parts. Once this is done, each cross-term may be implemented using one real filter chip and the proper filter outputs summed together to form two outputs which may be treated as a complex number pair;

$$c_k = a_k + jb_k \qquad \text{and} \quad y(nT) = \sum x(nT - k)\ c_k \qquad (3\text{-}21)$$

$$\mathrm{Re}\{y(nT)\} = \sum \mathrm{Re}\{x(nT - k)\}\ a_k - \mathrm{Im}\{x(nT - k)\}\ b_k \qquad (3\text{-}22)$$

$$\mathrm{Im}\{y(nT)\} = \sum \mathrm{Im}\{x(nT - k)\}\ a_k + \mathrm{Re}\{x(nT - k)\}\ b_k \qquad (3\text{-}23)$$

The solution for these equations will be shown in the section on coefficient selection. Additional comments regarding implementation details may also be found in that section.

Inmos IMS A100 digital signal processing chips were used as the major filter components in the pulse compression filter. These chips are thirty-two stage, cascadable transversal filters. (By cascading these chips, filters longer than 32 taps may be built.) These chips implement a modified transversal filter architecture as shown in (Figure 10)[6]. The user's model of the A100 is shown in (Figure 11). Only those features used while implementing the pulse compression filter will be discussed.

As may be seen from (Figure 11), the A100 has two primary busses; the microprocessor interface buss and the dedicated data buss. The microprocessor buss is asynchronous. It has a sixteen bit wide data path, seven address lines, and the necessary control lines to allow interfacing to any general purpose sixteen bit microprocessor. The chip is addressed on word boundaries only. It is through this buss that the control and data registers of the chip are accessed during initialization of the system.

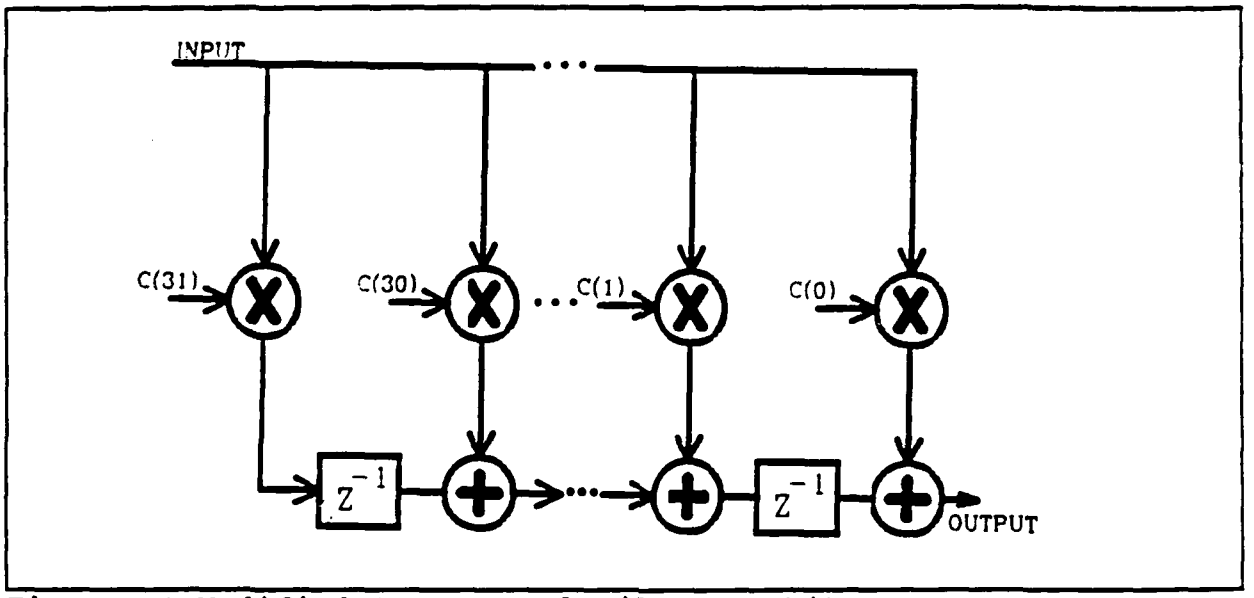

Figure 10 Modified Transversal Filter Architecture

The dedicated data buss is a high speed synchronous buss; the CLK and GO signals establish the timing on this buss. Three data ports comprise the balance of this buss system. These are the Cascade In port, Data In port, and the Data Out port; the Data In port is sixteen bits wide while the Data Out and Cascade In ports are twenty-four bits wide (twelve physical bits, time-multiplexed). The Data Out port provides the filtered output. This data is strobed with both edges of the OUTRDY signal; most significant word (MSW) on the rising edge and least significant word (LSW) on the falling edge. When used in a cascade, the Data Out port of the upstream chip is connected to the Cascade In port of the downstream chip and filtered data is obtained from the downstream chip. When used in a multi-chip cascade, the Data In ports of all devices shall be parallelled.

The CLK signal provides internal timing for the chip's Multiply Accumulate array (MAC). The GO signal, in this application, is an input to the chip; it essentially functions as a data valid strobe. The general timing relationships on this buss are shown in (Figure 12). For more details, see the A100 data sheet.

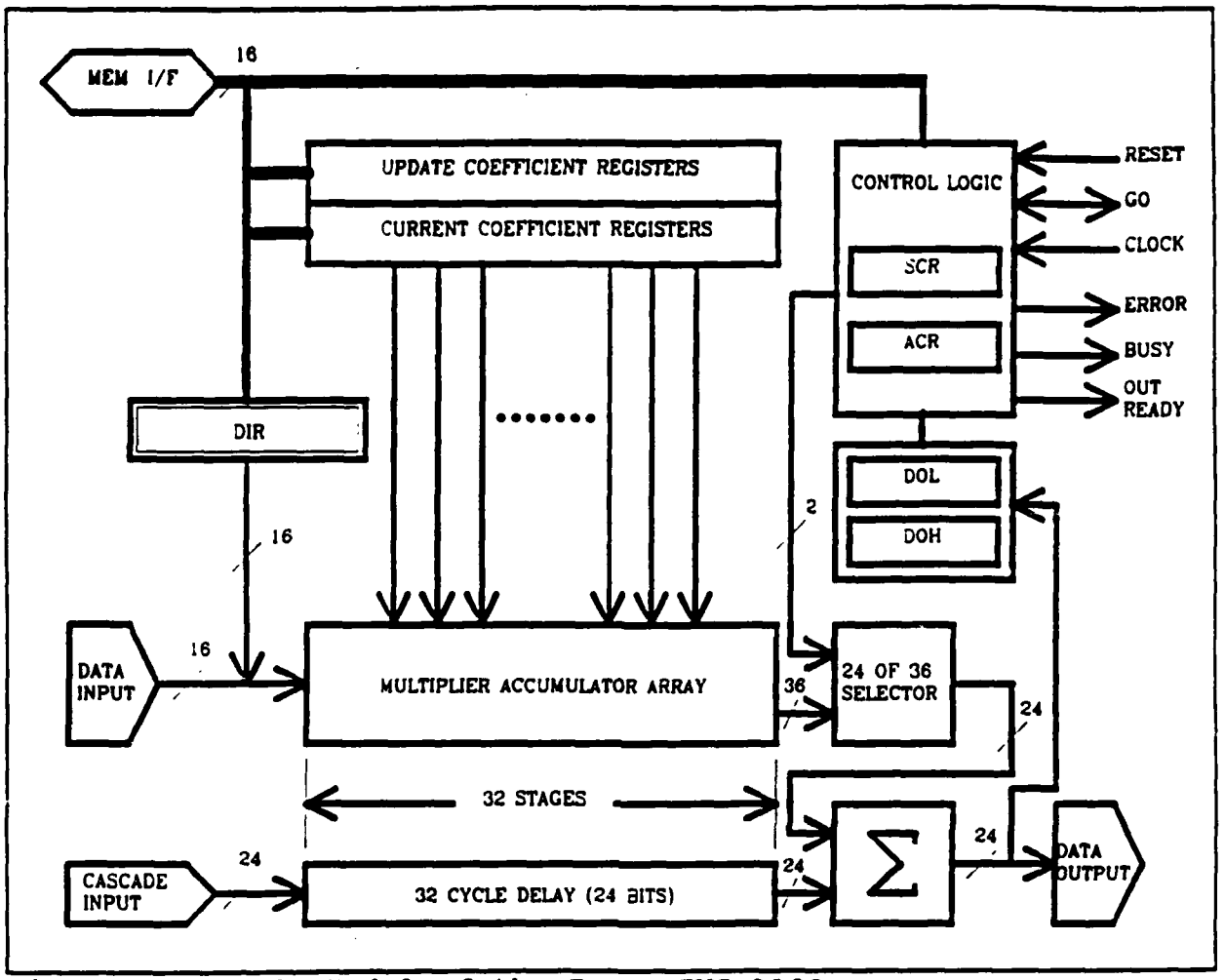

Figure 11 User's Model of the Inmos IMS A100

The A100 has two types of registers. These are the control registers and the coefficient registers. There are three control registers in the A100: Static Control Register (SCR), Active Control Register (ACR), and Test Control Register (TCR). The SCR's bits determine the mode of operation of the chip. [To ensure data integrity, all chips in a cascade must operate in the same mode.] The ACR functions as the overflow error indicator for the pulse compressor. The TCR has a bit in it that allows entering the Test Mode. This mode allows the least significant bits of the accumulator to be examined. Test mode also overrides error indication in the ACR as well as some SCR settings. Operating in Test mode is required when only one or two array elements are on.

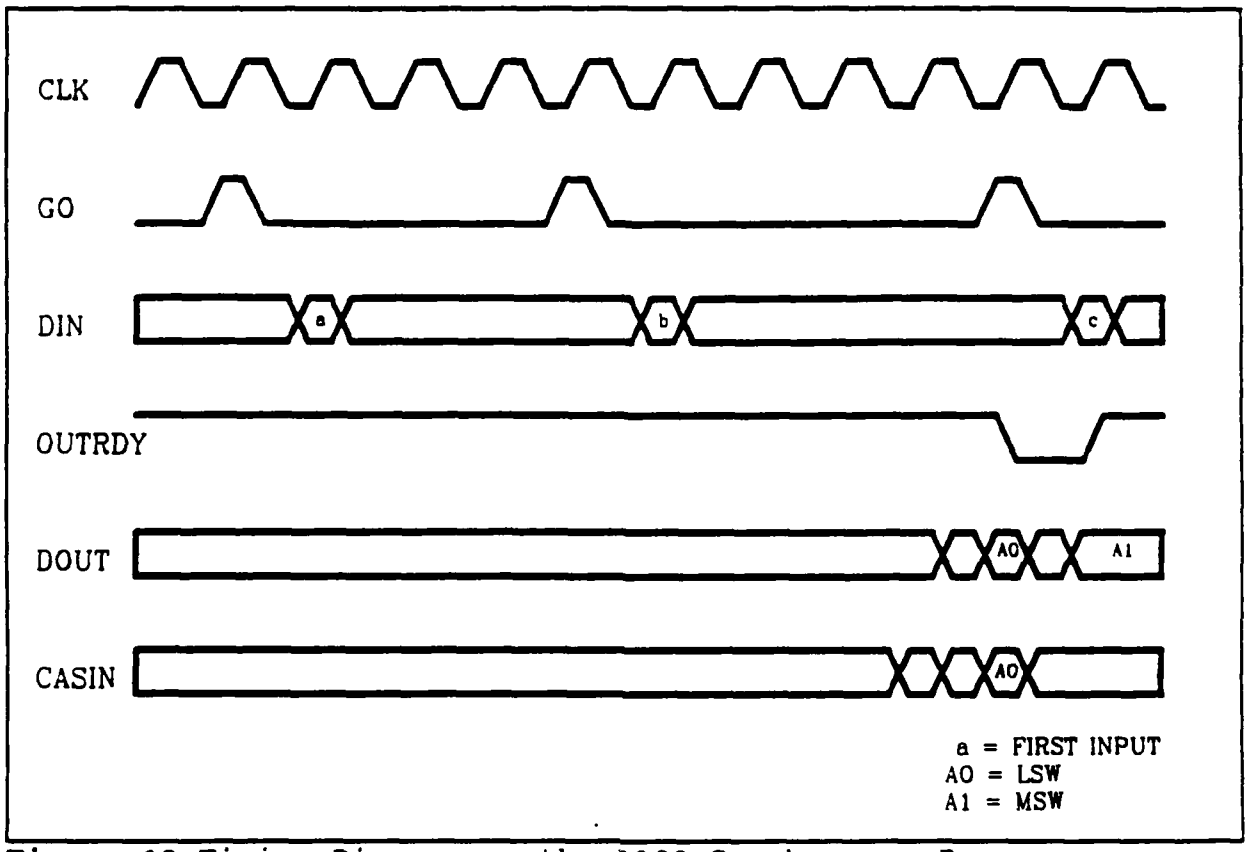

Figure 12 Timing Diagram on the A100 Synchronous Bus

The SCR controls four parameters significant to the pulse compressor. These are coefficient size, output range, fast output and slave modes. Coefficient size is set to eight bits which yields a 5Mhz throughput rate. The throughput rate may be increased at the cost of coefficient size. The output range determines which bits of the 36 bit MAC are sent to the 24 output bits. The pulse compressor uses the right-hand most range available. This translates into the MAC bits [30-19] mapping into output bits [11-0]. Remember, only the MSW is used. The Output range selection is overridden by invoking Test mode. Fast output mode ensures that the MSW is presented at the output as quickly as possible. Slave mode indicates that the input data will be synchronized to an externally generated GO signal. The GO signal is derived from Data Ready [DR] signals sourced by the complex beamformer circuitry.

The coefficient registers are organized as two, 32 word banks, each word being sixteen bits wide. The first bank is labeled Current Coefficient Registers (CCR) and the second Update Coefficient Registers (UCR). When writing coefficients to these registers, right hand justification and two's compliment notation are required. Only the CCR's are connected to the MAC array. The UCR's may store an alternate coefficient set. The contents of the two banks may be exchanged by setting the Bankswap bit in the ACR.

The internal memory map of the A100 is shown in (Figure 13). Notice that all addressing is referenced to word boundaries. Also, notice that no information

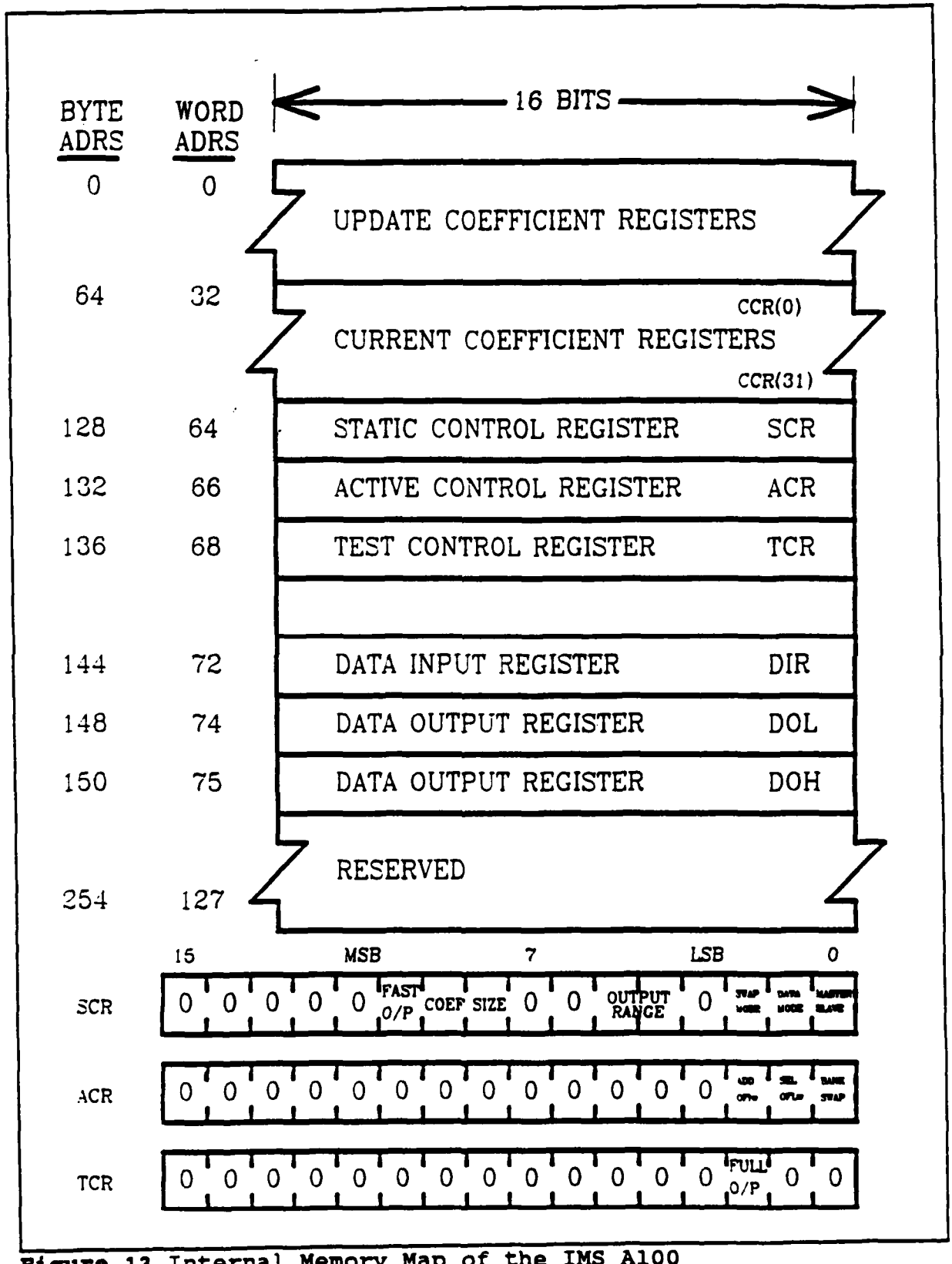

Figure 13 Internal Memory Map of the IMS A100

is given about the contents of locations above seventy-five. These locations are reserved and shown not be used because , at present (version C), internal decoding is not unique in this region.

The pulse compression filter is built on two boards, one board for each filtered output channel (I and Q). Each board is identical, both physically and functionally. (See Figure 14.) A filter board has two input data paths, one for each unfiltered I & Q stream. Each data path is made up of a cascade of two IMS A100 chips. The most significant (MSW) output word from each downstream chip is latched and then routed to the cross-term adder. The adder's output is fed to the "B" inputs of the 2 stage mux assembly. The "A" inputs of the mux are driven by unfiltered input streams; the "I" input drives the stage one mux and the "Q" input drives the stage two mux. The tri-state outputs of each mux stage are paralleled and fed to the output latches. These latches buffer the data delivered to the fast memories.

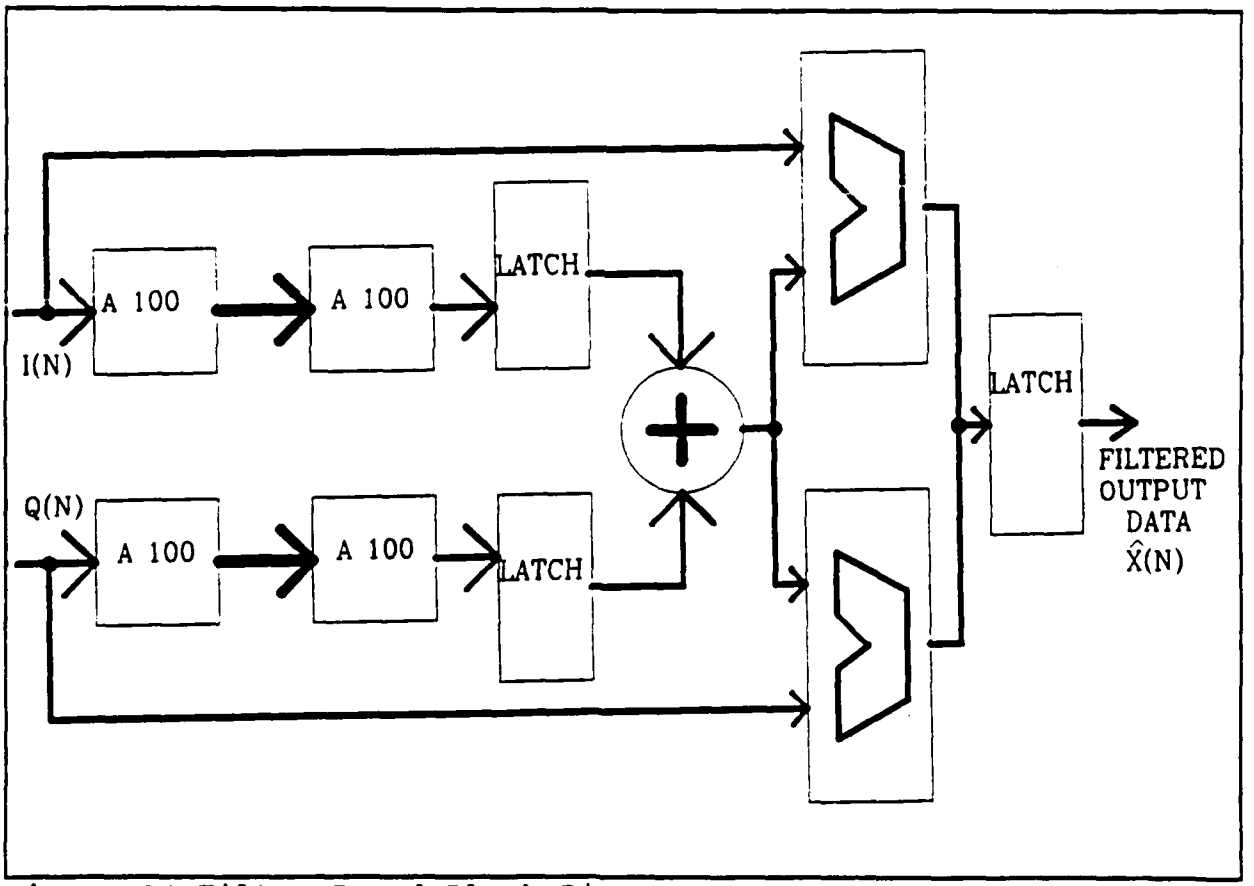

Figure 14 Filter Board Block Diagram

The mux assembly provides a bypass mode of operation for the filter. A control input [FILTER/BYPASS*] on the rear of the chassis selects either filtered or raw data. Another control input, hardwired at each card slot, tri-states one of the mux stages. This signal is wired so that only the stage one (or stage two) mux's output is enabled on the card in the "I" (or "Q") slot. This feature allows the boards to be slot independent.

The memory map of a filter board is shown in (Figure 15). The filter chips on a board are mapped into four contiguous 256 byte locations. The chips were mapped in this manner to exploit the "indirect with offset" addressing mode available on today's microprocessors. (Figure 16)[7] shows the mapping of filter coefficients, with canonical indices, into a cascade of A100s with their respective indices. The A100 indices are shown in hexidecimal notation because the control program (monitor) requires this format.

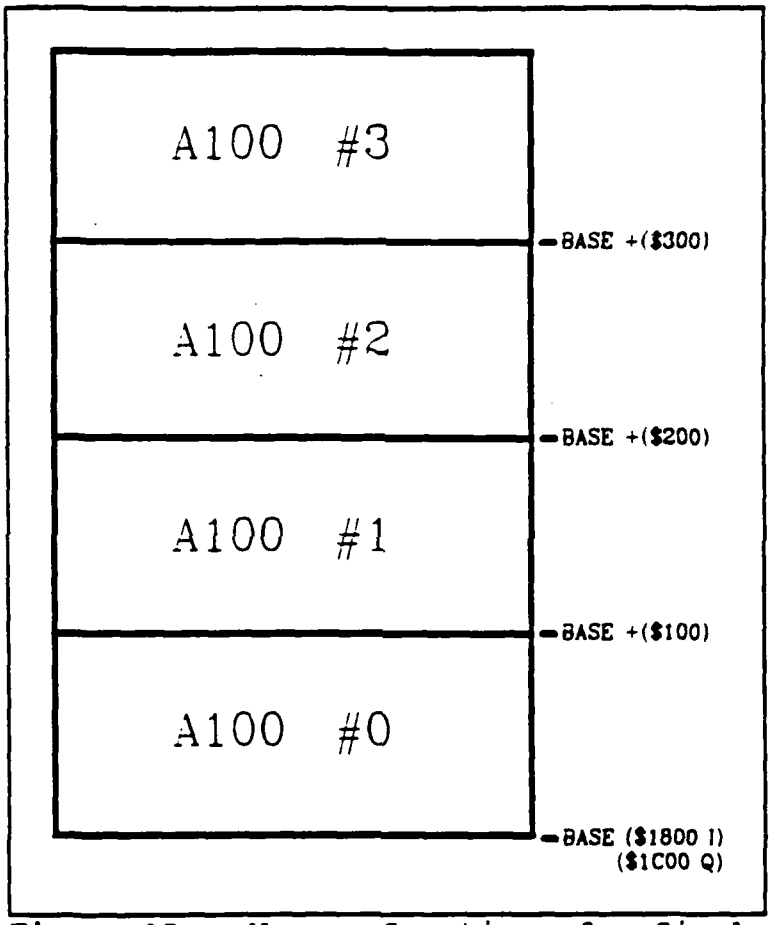

Figure 15 Memory Location of a Single Filter Board

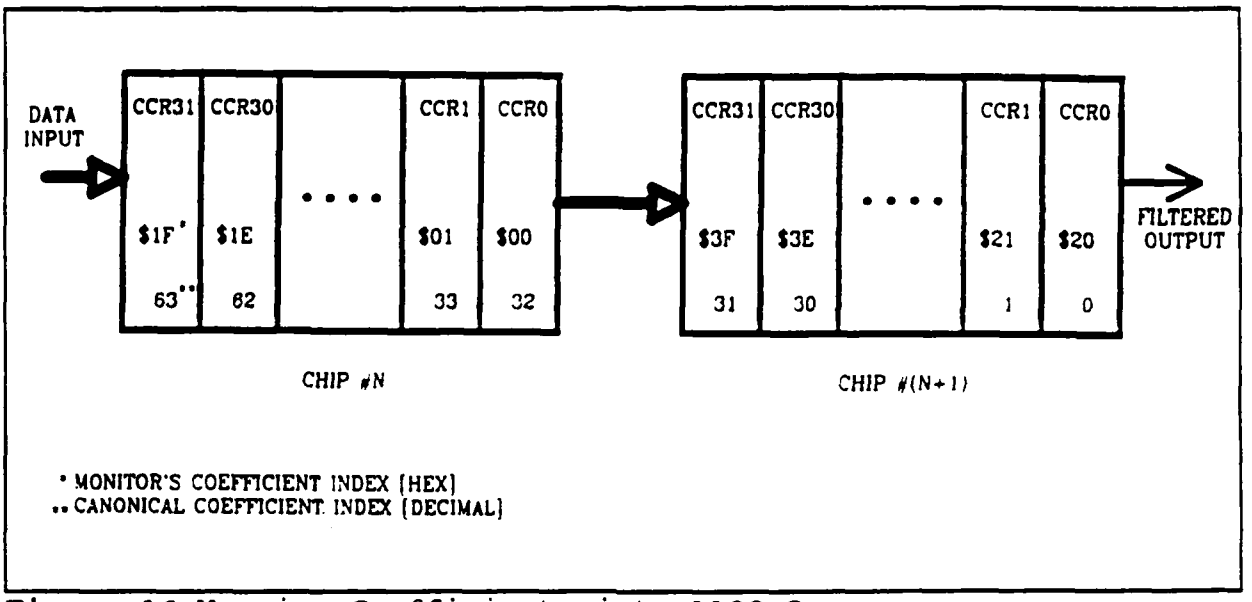

Figure 16 Mapping Coefficients into A100 Space

III D. MICROPROCESSOR BASED FILTER CONTROLLER

The pulse compression filter circuitry is controlled by a Motorola MC68000 microprocessor based filter control card. This processor, along with its supporting peripherals, oversees initial configuration of the IMS A100s, monitors their operation as they filter data, and provides the necessary link to allow user interactions. The remainder of this section will discuss the hardware aspects of the filter controller. System Block Diagram is shown in (Figure 17).

The MC68000 has a vast number of facilities available to the system designer[8]. However, in this application, only those used are discussed. The processor runs in the Supervisor State instead of the User State so all available instructions may be used. The only exception vectors implemented are RESET and Level 1 & 2 Autovectors. Level 1 Autovector handles the interrupt from the Asynchronous Communications Interface Adapter [ACIA] when user input has been detected. Level 2 Autovector is initialized to handle A100 error generated interrupts.

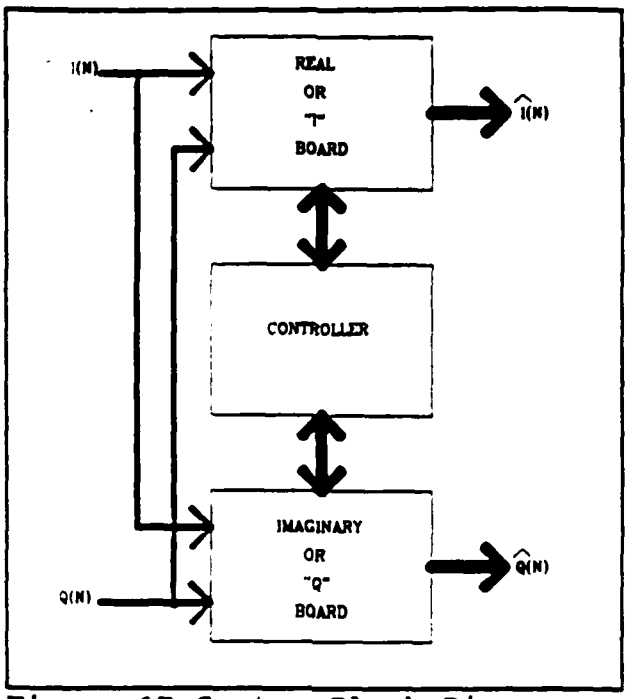

Figure 17 System Block Diagram

The memory map for the filter controller is shown in (Figure 18). Boundaries for the address decoder were set at 2K byte boundaries. This accommodates the space required for the A100's as well as provide sufficient general purpose I/O. Both EPROM and SRAM are decoded using two contiguous 2K byte outputs, thus providing 4K bytes of space in each region. Two spare decoder outputs are provided for future additions. The EPROM, which holds the executable code for the controller, resides at the bottom of memory space. The SRAM holds the system stack and a coefficient image. The system stack, being of the "push down" type, is located at the top of the decoded memory space while the coefficients are at the opposite end. A100 space occupies the slot just below the SRAM. The individual A100s are enabled by individual decoders on their respective boards. The ACIA is the only device which occupies GPIO space. Since it is an eight bit device, it can only be placed on one half of the data bus. The odd half was chosen to minimize buss wire length. Decoding of the ACIA in this region is not unique.

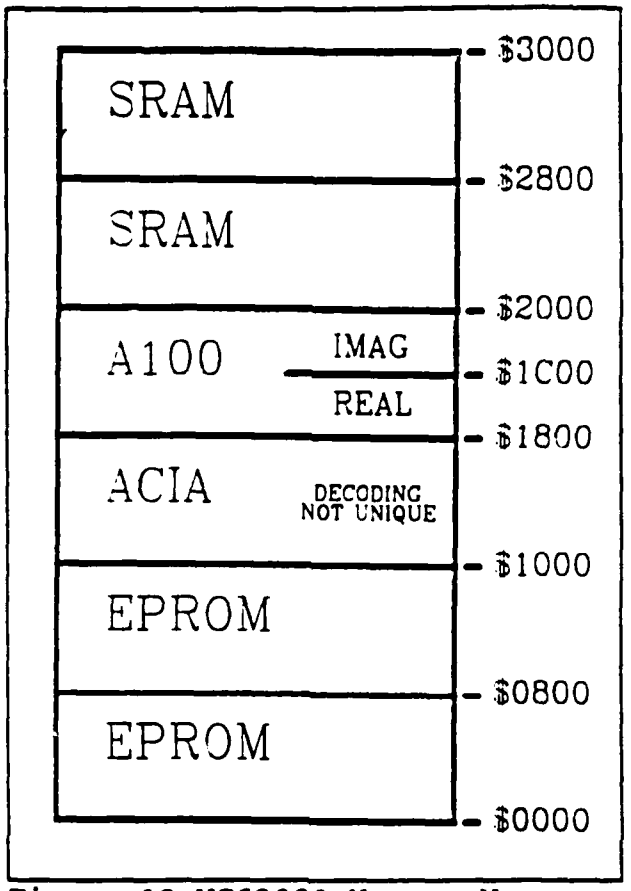

Figure 18 MC68000 Memory Map

All of the processor's peripheral devices are connected to the asynchronous bus with the exception of the ACIA, which is connected to the synchronous bus. Since the synchronous bus was incorporated into the 68000 for devices of this type, interfacing is straight forward[9]. The asynchronous bus is not as simple though because it requires a DTACK* signal from each device, acknowledging transfer of data, to end a bus cycle. With the processor running a five Mhz, the A100's cycle time is fast enough to allow its CE* signal to function as its DTACK* source. The EPROMs, on the other hand, are too slow to use this technique. This problem is handled by enabling a counter with the EPROM's CE* signal. After 200 ns, this counter generates the DTACK* signal and resets itself. Though the SRAMs are fast enough not to require this circuit, they are connected to the counter to minimize memory interface circuitry.

The serial port provides the physical connection between the controller and the user terminal. The voltage levels are compatible with EIA RS-232-C standard. There are no provisions in this interface for hardware handshaking; i.e.the minimal three wire terminal connection is implemented (TX DATA, RX DATA, and SIGNAL GROUND). Input data from the user should be seven bit ASCII data sent at 9600 baud with even parity. Upon reception of a valid character, the ACIA interrupts the processor via the Level 1 Autovector. The exception [interrupt] handler validates the interrupt and places the character in the Status Register, D0, for the program. Since maximum data string length (input or output) was

determined to be less than eighty characters, the XON/XOFF software protocol was omitted.

The MC68000 controller software was written in assembly language, using modular routines. This technique has two important advantages; it allows one block of code to be used by several procedures and it minimizes the effort required when adding new functions to the program's monitor loop. The controller program performs initialization of the pulse compression hardware as well as monitoring its operation. A flowchart showing the major features of the Controller program is shown in (Figure 19). The details of these functions are discussed in the next two sections.

The initialization process for the filter is always performed upon a power up or reset. First, the ACIA is initialized so that the terminal interface will function. Then the filter chips are initialized. Following this action a "hello" message is sent to the terminal followed by a display of the operator's menu. Finally, the monitor loop is entered. In this loop, the processor continually checks the status of the filter chips waiting for either a filter chip error or an interrupt from the user.

The ACIA initialization is straight-forward. A reset control word is sent to the device twice followed by the mode word. The mode of operation chosen is 16X clock, seven bit data, one stop bit and even parity.

The filter chips' initialization process begins by writing the desired mode words to the appropriate control registers. These registers are the Static, Active & Test Control registers. The Static Control Register value used selects 8 bit coefficients, a "fast" output mode, and dedicated I/O ports for the data to be filtered. The Active Control Register functions like a Status Register, its initial value is zero. The TCR's initial value selects the Test mode. Selecting the Test mode gives the filter chips a gain factor of sixteen and allows small signals to be readily detected. Once the control registers have been set up, the coefficients are ready to be loaded into the chips.

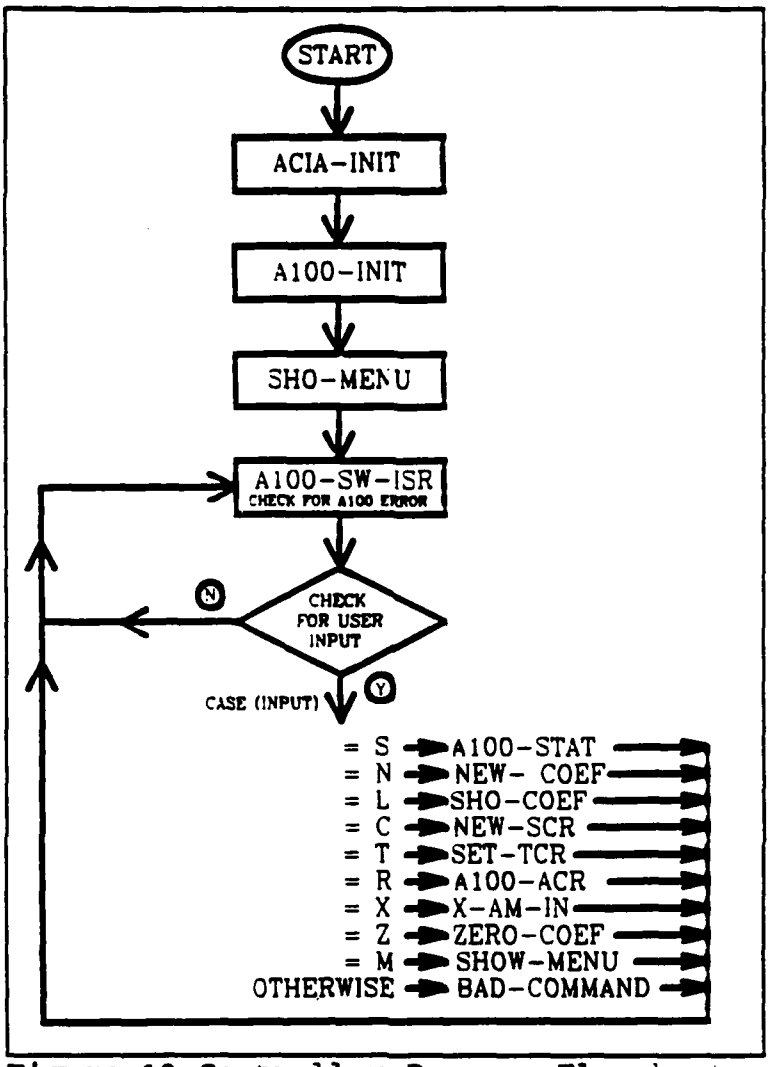

Figure 19 Controller Program Flowchart

Default coefficients are stored in EPROM so that they are readily available at boot time. They are written to the DSP chips in a two step process: 1) copy the coefficients from EPROM to RAM, and 2) copy the coefficients from RAM to the filter chips. This two step process allows the code that performs the actual RAM-to-DSP chip operation to be reused later for modifying the coefficients.

Now that the coefficients have been written to the DSP chip, the boot routine sends a "hello" message to the terminal and then displays the user interface menu. This is accomplished by recurring use of a SEND_MESsage routine. An address register, A1, functions as a pointer to a variety of ASCII message strings stored in the EPROM. The SEND_MESsage routine sends the message string pointer in register A1 to the terminal.

After the menu is sent out to the terminal the program enters the monitor loop. The microprocessor executes two functions while in this loop. The first is to check for an error from the DSP chips. This operation is performed by a subroutine that checks all of the DSP chips ACR for a non-zero; i.e. error,

condition. If no error exists, control of the program returns to the loop. However, if an error does exist, a message displaying the error is sent to the operator and an error handling routine is invoked to reset the error condition. When the error is eliminated, program control returns to the monitor loop.

The second function of the monitor loop is to check the program's status byte - Register D0. This byte is used as the communication link between the user and the program. As long as the status byte is equal to zero, no operator request has occurred and the program execution continues at the top of the loop. If the status byte indicates a non-zero condition, a user request is indicated. A series of IF_THEN_ELSE statements are executed to determine the request. Once the request is identified, the appropriate subroutine is invoked.

The user request is a single ASCII character, deposited in the Status Byte via the ACIA, sent by the user. Valid requests are C, L, M, N, R, S, T, X, and Z. The tasks performed when these requests are detected are listed below:

C -Change SCR value of output field selection, functions as gain control when not in Test Mode.

L -List the filter coefficients from a particular DSP Chip.

M -Display available Menu functions.

N -Enter New DSP coefficients.

R -Reset the ACRs on the DSP Chips.

S -Send status of DSP Chips to Terminal.

T -Enter/Exit Test mode.

X -Examine a specific word location in the DSP CHIP ADDRESS SPACE.

Z -Zero <u>all</u> Filter coefficients.

If an invalid command is detected, an error message is sent to the user and the monitor loop is restarted. New functions are easily added to the program by writing the "service routine" required by the new function and inserting an IF_THEN_ELSE type of statement in the code so the program will recognize the new function when it is issued.

III. E. SYSTEM INTEGRATION

Two sets of coefficients for the pulse compression filters have been obtained using the theory presented earlier. The first set of coefficients were

obtained from a purely mathematical model of an FM chirp pulse obtained from Cook's paper. The second set of coefficients were obtained from an actual sample of a transmitted pulse. Initially, no windowing techniques were employed when both coefficient sets were generated so relatively high (13 dB) sidelobes were expected. The remainder of this section explains how the coefficient values were determined.

To derive an expression for the coefficients of the pulse compression filter, first consider the received pulse, x(t). It may be represented as:

$$x(t) = \exp\ [j(ut^2/2)] \qquad |t| <= T_o \tag{3-24}$$

where u characterizes the chirp encoded in the pulse and T_o is one half of the pulse width. Its Fourier transform is of the form:

$$X(\omega) = \exp\ [-j(\omega^2/2u)]. \tag{3-25}$$

Substituting the complex conjugate of this expression into the one obtained earlier for the filter coefficients yields:

$$c_k = 1/2 \int \exp[j(\omega^2/2u - \omega kT)]\ d\omega. \tag{3-26}$$

By completing the square in the exponential, the expression becomes:

$$c_k = 1/2 \quad \exp[\ -ju(kT)^2] \int \exp[j(\omega/\ 2u - 2u/2(kT)^{)2}]\ d\omega. \tag{3-27}$$

The integrand of this expression is of the form $\exp[j(z^2)]$, which is a form of the Fresnel integrals[10]. By a substitution of variables and the evaluation of this integral, it may be shown that the integral contributes a complex constant common to all coefficients. Since these coefficients will be scaled, any terms common to all coefficients may be ignored. Thus, the expression for the c_k's is now:

$$c_k = \exp\ [\ -j(u(kT)^2/2)]. \tag{3-28}$$

Breaking this expression into its components gives expressions for the coefficients which are easily evaluated on any computer. (The program used to generate the coefficients used is given in the Appendix.) These expressions are:

$$a_k = \cos\ [\ u(kT)^2/2\ ] \tag{3-29}$$

$$-b_k = \sin\ [\ u(kT)^2/2\ ]. \tag{3-30}$$

Digital processing may be carried out in either fixed or floating point math. Fixed-point devices are usually faster and cheaper than their floating-point counterparts; integer math's major limitation is lack of dynamic rangel. In many cases, this one in particular, fixed point implementation is adequate. The precision limit imposed on the coefficients is eight bits (two's complement form). The representation may represent a maximum magnitude of 127. Since the maximum value of the a_k's and b_k's never exceeds unity, the scale factor is set to 127 to make full use of the dynamic range provided. For further details regarding FIR filter design techniques, see the reference list.

Special I/O Timing Requirements

The GO signal is an externally generated signal required by the A100 which synchronizes the data input process. Data is guaranteed to be valid for 150 ns after the rising edge of DR. DR, from the digital beamformer circuitry, is a positive pulse approximately 50 ns wide with a nominal period of 200 ns. The A100 senses its GO input on the rising edge of every CLK (20 Mhz) for a valid GO signal. Once a valid GO is detected, the Data In port is sampled on the next rising edge of CLK. With eight bit coefficients, a new sample of data is accepted every 4 CLKs [at 5Mhz] provided a valid GO signal is sensed. The DR signal must be synchronized to the CLK signal in order for it to function as the GO signal. This is accomplished by inverting the DR signal to obtain DR^*. This yields a 75% duty cycle vs. a 25% duty cycle. The DR^* signal passes thru a latch clocked by CLK. This ensures that the setup and hold requirements placed on the GO input are met. It should be noted that due to the 75% duty cycle, sometimes the latched DR^* signal, GO, stays high for several data input cycles. This does not violate the GO timing parameters because GO never has to return to the low state once it is asserted.

IV. SOFTWARE DEVELOPMENT

The software developed for the adaptive beamforming system can be logically subdivided into two parts. The first part is the adaptive weight control and data aquisition program. This software is designed to run in a real-time system with the data acquisition allowing for offline analysis. The second part is a collection of analysis programs for determining system performance and various system parameters, such as the optimum compression weights for the digital matched filter.

Referring back to the introduction, the digital beamformer was designed to overcome the IF beamformers inability to properly evaluate open loop adaptive algorithms without sacrificing the ability to evaluate closed loop adaptive algorithms. In order to test the digital beamformers ability to implement both types of algorithms, the original software written for the IF beamformer was modified by replacing the weight output device driver and modifying the data acquisition subroutine to account for the fixed pipeline delay of the new digital system. As a result of the vastly improved speed and accuracey of the new output weight device driver, the data acquisition code was able to run over an entire scan of 0 to 180 degrees. The offline analysis was then capable of examing the beamformers quiescent and adapted performance over an entirs scan. This was not feasible with the IF beamforming system due to the time required for just one beam angle.

V. ANALYSIS

The performance of the adaptive beamforming system was evaluated both for its ability to suppress interference sources as well as the pulse compressors ability to provide for increased range resolution with a specified sidelobe level of 30 dB. Because the performance of each of the two primary subsystems does not effect each other, their analysis was handled separately.

First the performance of the adaptive beamformer was tested using a technique that normalizes the quiescent, unadapted, response to the maximum ratio of the interference noise power to the desired signal power over all scan angles. With one interference source this maximum occurs when the quiescent, unadapted, weights of the beamformer have the beam steered in the direction of the interference source.

The program progressively scans the main beam from 0 to 180 degrees (θ), in increments of 1 degree. The unadapted response of the desired signal in the main beam is measured in the absence of sidelobe interference ($S_u(\theta)$). Next, the unadapted response of the sidelope interference is measured in the absence of the desired signal($J_u(\theta)$). The program then runs an adaptive algorithm and computes the optimum weighting vector for this particular look angle. The beamformer's weights are set to this optimum weight vector and the above mentioned measurements are made again for the adapted response ($S_a(\theta)$ and $J_a(\theta)$). This data is store to magnetic tape for further offline processing.

The data is uploaded to a general purpose computer for its final processing and graphical display. Two curves are calculated, the first being the quiescent steered response. This curve is simply:

$$\text{QUIESCENT} = J_u(\theta)_{dB} - S_u(\theta)_{dB} - F \qquad (5\text{-}1)$$

where F is the normalization factor first mentioned. The second curve is the *adapted steered response:*

$$\text{ADAPTED} = J_a(\theta)_{dB} - S_a(\theta)_{dB} - F \qquad (5\text{-}2)$$

These two curves are plotted together as a function of angle. Figure 20 is the result of this process using the digital version of the Howells_Applebaum adaptive algorithm[11], a closed loop or feed back algorithm. This particular case was run with the Surface Acoustic Wave (SAW) filters still present in the system.

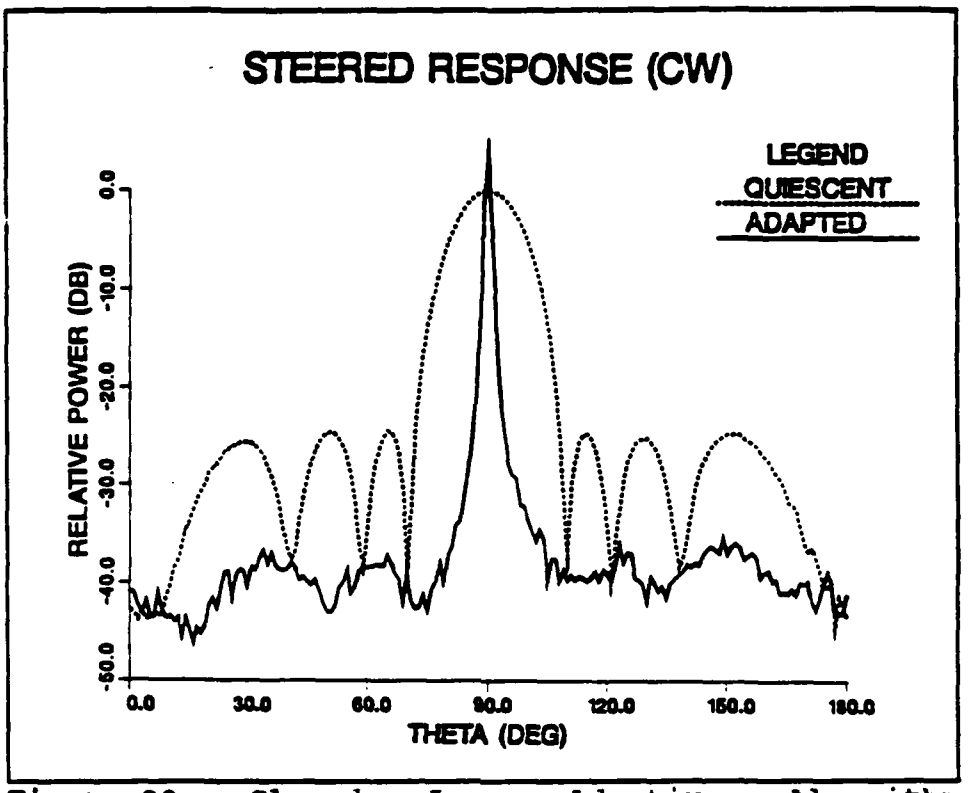

Figure 20 Closed Loop Adaptive Algorithm (Howells-Applebaum)

Figure 21 is the result of using the Sampled Matrix Inverse algorithm[12], an open loop adaptive algorithm. This case was run with Lump Constant (LC) filters present in the system. The LC filters have a much better channel to channel match than the SAW filters delivering an average 6 dB improvement in the sidelope suppresion of the open loop algorithm compared to the closed loop algorithm.

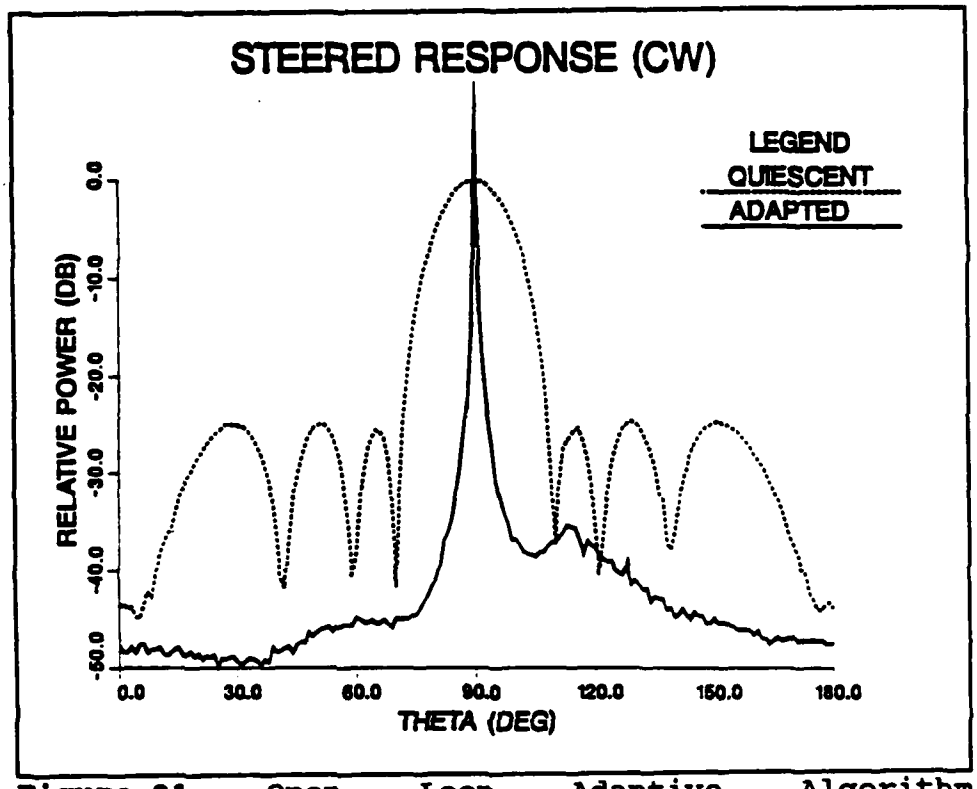

Figure 21 Open Loop Adaptive Algorithm (SMI)

Performance of the pulse compression filter was analyzed based upon three criterion. These are the group delay, time sidelobe levels, and 3 dB pulse width. The first performance value is easily measured by comparing the pre and post compressed pulses. Figure 22 reveals that the center of the uncompressed pulse is 4 μsec and that the center of the compressed pulse is 11.8 μsec. The delta time difference is 7.8 μsec. Finally, the last two performance values, the time sidelobe levels and the 3 dB pulse width, can be measured from Fig. 23. The sidelobe level is less than 30 dB below the relative perk amplitude of the pulse and the 3 dB pulse width is 600 nsec. This corresponds to the bandwidth of the LC filters and is the best that can be expected with these filters. The range resolution can be increased by using wider bandwidth LC filters up to a maximum of 5 Mhz at which time the A/D converters and synchronous clock speed of the digital filter will begin to alias and degrade signal quality.

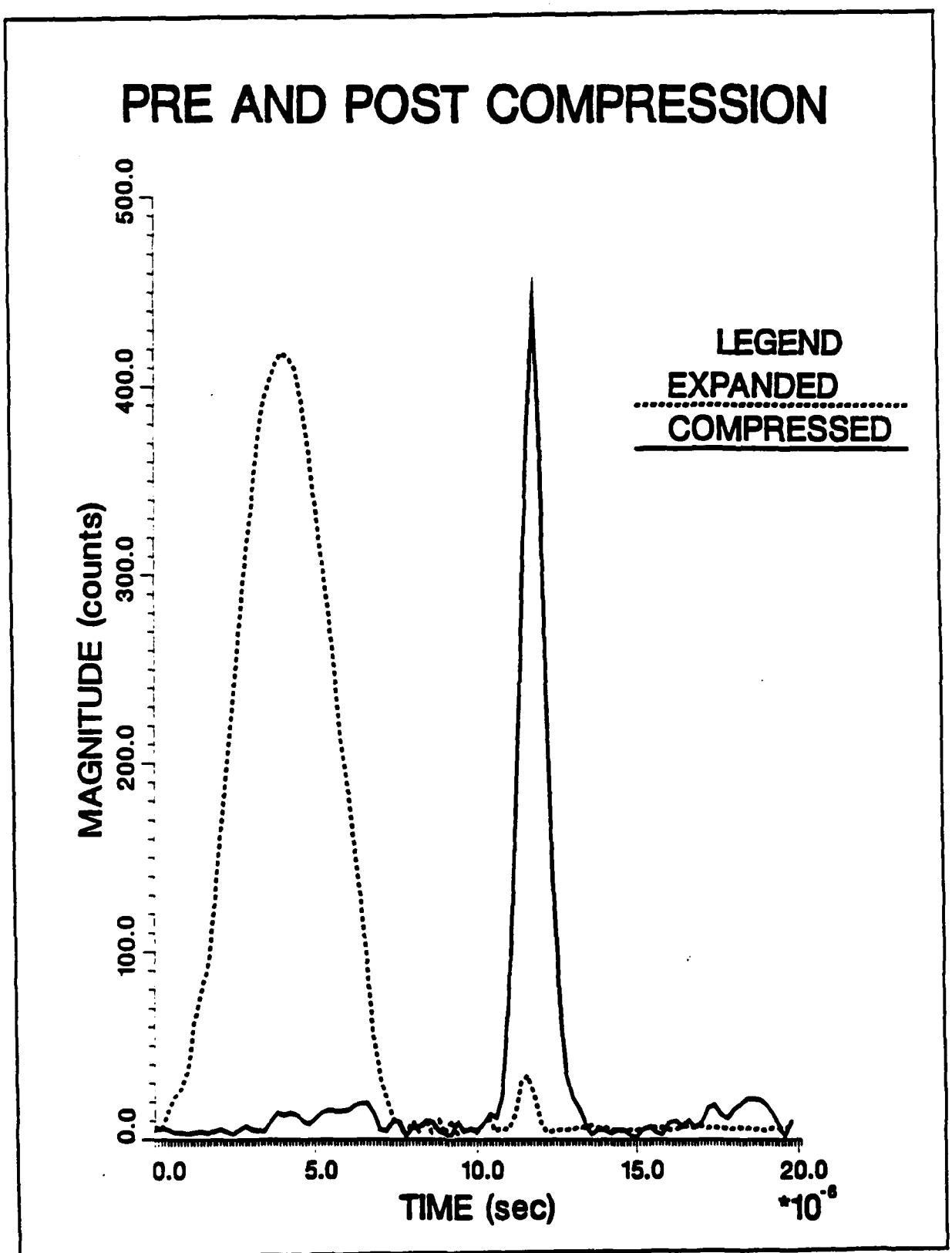

Figure 22 Pre and Post Compression Relative Timing

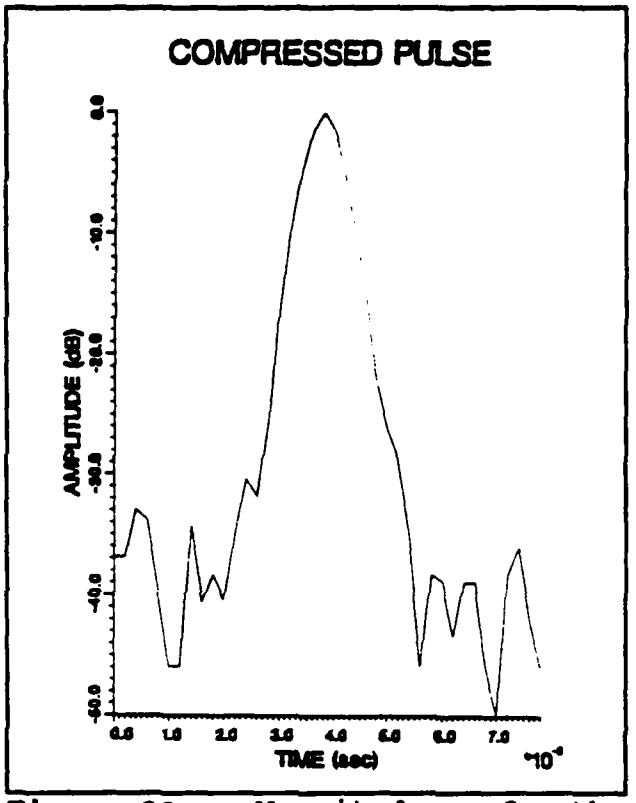

Figure 23 Magnitude of the Compressed Pulse

VI. SUMMARY

This report detailed the addition of a digital beamforming system with digital pulse compression to the Adaptive Processing Laboratory and described the capabilities of the system in implementing and testing various adaptive algorithms. The digital beamforming system replaced two components, the IF beamformer and the SAW compression filters, that were impeding the use of open loop adaptive algorithms and degrading system performance. The new system is capable of implementing both closed loop and open loop adaptive algorithms with the same precision and accuracy.

VII. RECOMMENDATION

The digital beamforming system is ideally suited to performing both adaptive and superresolution algorithms and measuring the performance of such algorithms in a real-time system. However, the computational requirements of modern algorithms has progressed beyond the current capabilities of the current real-time adaptive processor. Also, the most advanced superresolution algorithms require a high degree of channel matching that currently is not available in the Adaptive Processing Laboratory. It is therefore recommended that the real-time adaptive processor be upgraded and that a preprocessor to the digital beamforming system be added to allow for real-time digital calibration of the incoming data sampled data.

VIII. REFERENCES

1. Lee, F. W., "Development of a Hybrid Adaptive Beamformer", NRL Report 9249, February 28, 1990

2. Analog Devices Inc., "Analog Devices 1984 Data Book: Volume I Integrated Circuits", Analog Devices, Norwood, MA., 1984 pp. 12-28

3. Cook, C.E., "Pulse Compression: Key to More Efficient Radar Transmission", Proceedings IRE, vol. 48, pp.310-315, March 1960

4. Stanely, W.D., Dougherty, G.R., Dougherty, R., Digital Signal Processing, Reston Publishing Company, Reston, VA., 1984

5. Oppenheim, A.V., Willsky, A.S, Young, I.T., Signals and Systems, Prentice Hall Inc., Englewood Cliffs, NJ., 1983

6. Inmos Corporation, "IMS A100 Data Sheet (Preliminary)", Inmos Corporation, Colorado Springs, CO, 1986

7. Yassie, H., "IMS A100 Application Note 1: Digital Filtering with the IMS A100" and "IMS A100 Application Note 3: Correlation and Convolution with the IMS A100", Inmos Corporation, Colorado Springs, CO, 1986

8. Motorola Semiconductor Products Inc., "MC68000 16/32 Bit Microprocessor Reference Manual", Motorola Inc., Austin, TX., 1985

9. Motorola Semiconductor Products Inc., "MC68000 16/32 Bit Microprocessors Programmer's Reference Manual", Motorola Inc., Austin, TX., 1985

10. Abramowitz, M., Stegun, I.A., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, National Bureau of Standards, Washington D.C., 1967

11. Applebaum, S. P., "Adaptive Arrays", IEEE Transactions on Antennas and Propagation, vol. AP-24, pp. 585-598, September 1976

12. Monzingo, R., and Miller, T., Introduction to Adaptive Arrays, Wiley-Interscience, Ney York, NY., 1980, Ch.6